Análisis de complejidad algorítmica

Merge Sort:

|  |  |  |
| --- | --- | --- |
| Ordenamiento Merge | Costo | # veces que se ejecuta |
| if (arr.Length < 2) { | C1 | Flor(n/2) |
| return arr;  } | C2 | Flor(n/2)-1 |
| int[] left; | C3 | 1 |
| int[] right; | C4 | 1 |
| int mid = arr.Length / 2; | C5 | 1 |
| left = new int[mid]; | C6 | 1 |
| if (arr.Length % 2 == 0) | C7 | 1 |
| right = new int[mid]; | C8 | 1 |
| Else {  right = new int[mid + 1];  } | C9 |  |
| for (int i = 0;i<mid;i++){ | C10 | n/2 |
| left[i] = arr[i];  } | C11 | n-1 |
| int x = 0; | C12 | 1 |
| for(int i=mid;i<arr.Length; i++) { | C13 | n |
| right[x] = arr[i]; | C14 | n-1 |
| x++;  } | C15 | n-1 |
| left = MergeSort(left); | C16 | 1 |
| right = MergeSort(right); | C17 | 1 |
| return merge(left, right); | C18 | 1 |
| BigO |  | *n*log2​*n* |

Análisis analítico:

Un diagrama con un árbol a la izquierda y tiempos de mezcla a la derecha. El árbol está etiquetado como "Tamaño del subproblema" y la derecha está etiquetada como "Tiempo total de mezcla para todos los subproblemas de este tamaño".
El primer nivel del árbol muestra un solo nodo n y el tiempo correspondiente de mezcla de c por n. El segundo nivel del árbol muestra dos nodos, cada uno de 1/2 n, y un tiempo de mezcla de 2 por c por 1/2 n, que es lo mismo que c por n. El tercer nivel del árbol muestra cuatro nodos, cada uno de 1/4 n, y un tiempo de mezcla de 4 por c por 1/4 n, que es lo mismo que c por n.

Un diagrama con un árbol a la izquierda y tiempos de mezcla a la derecha. El árbol está etiquetado como "Tamaño del subproblema" y la derecha está etiquetada como "Tiempo total de fusión para todos los subproblemas de este tamaño".
El primer nivel del árbol muestra un solo nodo n y el tiempo correspondiente de mezcla de c por n. El segundo nivel del árbol muestra dos nodos, cada uno de 1/2 n, y un tiempo de mezcla de 2 por c por 1/2 n, que es lo mismo que c por n. El tercer nivel del árbol muestra cuatro nodos, cada uno de 1/4 n, y a tiempo de mezcla de 4 por c por 1/4 n, que es lo mismo que c por n. El cuarto nivel del árbol muestra ocho nodos, cada uno de 1/8 n, y un tiempo de mezcla de 8 por c por 1/8 n, que es lo mismo que c por n. Debajo de ese nivel, se muestran puntos para indicar que el árbol continúa de esa manera. Se muestra un nivel final con n nodos de 1 y un tiempo de mezcla de n por c, que es lo mismo que c por n.

Quieck Sort:

|  |  |  |
| --- | --- | --- |
| Ordenamiento QuickSort | Costo | # veces que se ejecuta |
| int pivot = Partition(arr, low, high); |  | S |
| QuickSort(arr, low, pivot - 1); |  | 1 |
| QuickSort(arr, pivot+1, high);  } |  | 1 |
| BigO |  | 3+S |

|  |  |  |
| --- | --- | --- |
| Partition | Costo | # veces que se ejecuta |
| int pivot = arr[high]; | C1 | Flor(n/2) |
| int lowIdx = (low - 1); | C2 | Flor(n/2)-1 |
| int[] left; | C3 | 1 |
| int[] right; | C4 | 1 |
| for (int j = low; j < high; j++) | C5 | nlogn |
| if (arr[j] <= pivot) | C6 | N - nlogn |
| lowIdx++; | C7 | N – nlogn -1 |
| int temp = arr[lowIdx]; |  | N – nlogn -1 |
| arr[lowIdx] = arr[j]; |  | N – nlogn -1 |
| arr[j] = temp; |  | N – nlogn -1 |
| int aux = arr[lowIdx + 1]; |  | 1 |
| arr[lowIdx + 1] = arr[high]; |  | 1 |
| arr[high] = aux; |  | 1 |
| return lowIdx + 1; |  | 1 |
| Big0 |  | **O(n\*log (n))** |